10/627,731

日本国特許庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application: ２００３年　７月１０日

出願番号
Application Number: 特願２００３－１９５１９４

[ST. 10/C]: [ＪＰ２００３－１９５１９４]

出願人
Applicant(s): 株式会社リコー

２００３年　７月２８日

特許庁長官
Commissioner,
Japan Patent Office 今井康夫

出証番号　出証特２００３－３０５９７８５

【書類名】　特許願
【整理番号】　0305290
【提出日】　平成15年 7月10日
【あて先】　特許庁長官　太田　信一郎　殿
【国際特許分類】　G03G 21/00　370
【発明の名称】　情報処理装置およびバージョンチェック方法
【請求項の数】　25
【発明者】
　【住所又は居所】　東京都大田区中馬込１丁目３番６号　株式会社リコー内
　【氏名】　秋吉　邦洋
【発明者】
　【住所又は居所】　東京都大田区中馬込１丁目３番６号　株式会社リコー内
　【氏名】　田中　浩行
【発明者】
　【住所又は居所】　東京都大田区中馬込１丁目３番６号　株式会社リコー内
　【氏名】　安藤　光男
【特許出願人】
　【識別番号】　000006747
　【氏名又は名称】　株式会社リコー
【代理人】
　【識別番号】　100070150
　【弁理士】
　【氏名又は名称】　伊東　忠彦
【先の出願に基づく優先権主張】
　【出願番号】　特願2002-224136
　【出願日】　平成14年 7月31日
【手数料の表示】
　【予納台帳番号】　002989
　【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【包括委任状番号】　9911477

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　情報処理装置およびバージョンチェック方法

【特許請求の範囲】

【請求項１】　アプリケーションと、前記アプリケーションからのＡＰＩを用いた要求に基づきシステム側の処理を行うシステムサービスとを備えた情報処理装置であって、

前記アプリケーションがシステムサービスに対して使用するＡＰＩのバージョン情報と、当該システムサービスが有するＡＰＩのバージョン情報を取得する取得手段と、

前記アプリケーションが前記システムサービスに対して使用するＡＰＩのバージョンを、ＡＰＩ単位に、前記システムサービスが有するＡＰＩのバージョンと比較する比較手段と

を有することを特徴とする情報処理装置。

【請求項２】　前記アプリケーションが前記システムサービスに対して使用するＡＰＩのセットのバージョンと、前記システムサービスが有するＡＰＩのセットのバージョンとを比較する手段を更に有し、

ＡＰＩのセットのバージョンが異なっている場合にのみ、前記比較手段による比較を行う請求項１に記載の情報処理装置。

【請求項３】　前記アプリケーションは、前記システムサービスに対して使用するＡＰＩのバージョン情報を、前記アプリケーションの実行プログラム中に備え、

前記取得手段が、当該ＡＰＩのバージョン情報を、前記アプリケーションから取得する請求項１に記載の情報処理装置。

【請求項４】　前記アプリケーションから前記バージョン情報を取得するために前記アプリケーションを仮起動させる請求項３に記載の情報処理装置。

【請求項５】　前記システムサービスは複数のシステムサービスを含み、

前記アプリケーションは、当該複数のシステムサービスに対して使用するＡＰＩのバージョン情報を、システムサービス毎に備え、

前記取得手段は、前記アプリケーションからあるシステムサービスに対応する

ＡＰＩのバージョン情報を取得した場合に、当該システムサービスから、当該システムサービスが有するＡＰＩのバージョン情報を取得する請求項３に記載の情報処理装置。

【請求項６】　前記アプリケーションが前記システムサービスに対して使用するＡＰＩのバージョン情報を格納したファイルを備え、

前記取得手段が、当該ＡＰＩのバージョン情報を、前記ファイルから取得する請求項１に記載の情報処理装置。

【請求項７】　前記比較手段による比較を前記アプリケーションのインストール前に行う場合において、

前記アプリケーションが前記システムサービスに対して使用する全てのＡＰＩのバージョンが、前記システムサービスが有する対応するＡＰＩのバージョンと一致する場合に、当該アプリケーションがインストール可能であることを表示する請求項１に記載の情報処理装置。

【請求項８】　前記情報処理装置は、前記情報処理装置におけるハードウェア資源の制御を行うコントロールサービスと、コントロールサービスをサーバとしたクライアントプロセスとして動作し、前記アプリケーションをクライアントとしたサーバプロセスとして動作する仮想アプリケーションサービスとを有し、

前記システムサービスは、アプリケーションからのＡＰＩを用いた要求を受信するコントロールサービスと、前記仮想アプリケーションサービスを含む請求項１ないし７のうちいずれか１項に記載の情報処理装置。

【請求項９】　前記仮想アプリケーションサービスは前記取得手段及び前記比較手段を有する請求項８に記載の情報処理装置。

【請求項１０】　アプリケーションと、前記アプリケーションからのＡＰＩを用いた要求に基づきシステム側の処理を行うシステムサービスとを備えた情報処理装置が実行するＡＰＩのバージョンチェック方法であって、

前記アプリケーションがシステムサービスに対して使用するＡＰＩのバージョン情報と、当該システムサービスが有するＡＰＩのバージョン情報を取得する取得ステップと、

前記アプリケーションが前記システムサービスに対して使用するＡＰＩのバー

ジョンを、ＡＰＩ単位に、前記システムサービスが有するＡＰＩのバージョンと比較する比較ステップと

を有することを特徴とするバージョンチェック方法。

【請求項１１】　前記アプリケーションが前記システムサービスに対して使用するＡＰＩのセットのバージョンと、前記システムサービスが有するＡＰＩのセットのバージョンとを比較するステップを更に有し、

ＡＰＩのセットのバージョンが異なっている場合にのみ、前記比較ステップによる比較を行う請求項１０に記載のバージョンチェック方法。

【請求項１２】　前記アプリケーションは、前記システムサービスに対して使用するＡＰＩのバージョン情報を、前記アプリケーションの実行プログラム中に備え、

前記取得ステップにおいて、当該ＡＰＩのバージョン情報を、前記アプリケーションから取得する請求項１０に記載のバージョンチェック方法。

【請求項１３】　前記アプリケーションから前記バージョン情報を取得するために前記アプリケーションを仮起動させる請求項１２に記載のバージョンチェック方法。

【請求項１４】　前記システムサービスは複数のシステムサービスを含み、

前記アプリケーションは、当該複数のシステムサービスに対して使用するＡＰＩのバージョン情報を、システムサービス毎に備え、

前記取得ステップにおいて、前記アプリケーションからあるシステムサービスに対応するＡＰＩのバージョン情報を取得した場合に、当該システムサービスから、当該システムサービスが有するＡＰＩのバージョン情報を取得する請求項１２に記載のバージョンチェック方法。

【請求項１５】　前記アプリケーションが前記システムサービスに対して使用するＡＰＩのバージョン情報を格納したファイルを備え、

前記取得ステップにおいて、当該ＡＰＩのバージョン情報を、前記ファイルから取得する請求項１０に記載のバージョンチェック方法。

【請求項１６】　前記比較ステップによる比較を前記アプリケーションのインストール前に行う場合において、

前記アプリケーションが前記システムサービスに対して使用する全てのＡＰＩのバージョンが、前記システムサービスが有する対応するＡＰＩのバージョンと一致する場合に、当該アプリケーションがインストール可能であることを表示する請求項１０に記載のバージョンチェック方法。

【請求項１７】　前記情報処理装置は、前記情報処理装置におけるハードウェア資源の制御を行うコントロールサービスと、コントロールサービスをサーバとしたクライアントプロセスとして動作し、前記アプリケーションをクライアントとしたサーバプロセスとして動作する仮想アプリケーションサービスとを有し、

前記複数のシステムサービスは、アプリケーションからのＡＰＩを用いた要求を受信するコントロールサービスと、前記仮想アプリケーションサービスを含む請求項１０ないし１６のうちいずれか１項に記載のバージョンチェック方法。

【請求項１８】　前記仮想アプリケーションサービスが前記取得ステップ及び前記比較ステップを実行する請求項１７に記載のバージョンチェック方法。

【請求項１９】　アプリケーションと、前記アプリケーションからのＡＰＩを用いた要求に基づきシステム側の処理を行うシステムサービスとを備えた情報処理装置に、

前記アプリケーションがシステムサービスに対して使用するＡＰＩのバージョン情報と、当該システムサービスが有するＡＰＩのバージョン情報を取得する取得手順と、

前記アプリケーションが前記システムサービスに対して使用するＡＰＩのバージョンを、ＡＰＩ単位に、前記システムサービスが有するＡＰＩのバージョンと比較する比較手順と

を実行させるプログラム。

【請求項２０】　前記アプリケーションが前記システムサービスに対して使用するＡＰＩのセットのバージョンと、前記システムサービスが有するＡＰＩのセットのバージョンとを比較する手順を更に実行させ、

ＡＰＩのセットのバージョンが異なっている場合にのみ、前記比較手順を実行させる請求項１９に記載のプログラム。

【請求項２１】　前記アプリケーションが前記システムサービスに対して使用するＡＰＩのバージョン情報を、前記アプリケーションの実行プログラム中に備え、

前記取得手順において、当該ＡＰＩのバージョン情報を、前記アプリケーションから取得する手順を実行させる請求項１９に記載のプログラム。

【請求項２２】　前記アプリケーションが前記システムサービスに対して使用するＡＰＩのバージョン情報を格納したファイルを備え、

前記取得手順において、当該ＡＰＩのバージョン情報を、前記ファイルから取得する手順を実行させる請求項１９に記載のプログラム。

【請求項２３】　前記比較手順における比較を前記アプリケーションのインストール前に行う場合において、

前記アプリケーションが前記システムサービスに対して使用する全てのＡＰＩのバージョンが、前記システムサービスが有する対応するＡＰＩのバージョンと一致する場合に、当該アプリケーションがインストール可能であることを表示する手順を実行させる請求項１９に記載のプログラム。

【請求項２４】　アプリケーションからの要求に基づきシステム側の処理を行うシステムサービスを備えた情報処理装置で実行される当該アプリケーションのプログラムであって、情報処理装置に、

前記システムサービスからの要求に基づき、仮起動するか通常起動するかを判断する手順と、

仮起動の場合に、前記システムサービスと通信することにより、アプリケーションに関する情報を前記システムサービスに提供する手順と

を実行させるプログラム。

【請求項２５】　請求項１９ないし２４のうちいずれか１項に記載のプログラムを記録したコンピュータ読み取り可能な記録媒体。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

この発明は、情報処理装置で実行されるアプリケーションと、情報処理装置の

システム間でのバージョン不整合による不具合発生を未然に防止することを可能とする技術に関する。

【０００２】

【従来の技術】

近年では、情報処理装置の１つとして、プリンタ、コピー、ファクシミリ、スキャナなどの各装置の機能を１つの筐体内に収納した画像形成装置（以下、「複合機」という。）が知られている。この複合機は、１つの筐体内に表示部、印刷部および撮像部などを設けるとともに、プリンタ、コピーおよびファクシミリ装置にそれぞれ対応した３種類のソフトウェアを設け、これらのソフトウェアを切り替えることによって、当該装置をプリンタ、コピー、スキャナまたはファクシミリ装置として動作させるものである。

【０００３】

このような従来の複合機では、プリンタ、コピー、ファクシミリ、スキャナなどの各機能単位でアプリケーションプログラムが起動され、ハードウェア資源にアクセスする機能を持ち合わせている。その際、アプリケーションプログラムが前提とするオペレーティングシステム（ＯＳ）と、実際のオペレーティングシステム（ＯＳ）のバージョンが同じことが前提となるが、例えば、ＯＳがバージョンアップして、アプリケーションとの間でバージョン差が生じた場合、今まで使えていた機能が使えなくなったり、アプリケーションそのものが起動しなくなったりすることがある。

【０００４】

このため、従来の複合機では、ＯＳがバージョンアップされた場合、それに伴ってアプリケーションをコンパイルし直すなどして、常に整合性のとれたバージョン関係にあることが要請されている。

【０００５】

【特許文献１】

特開２００２－８２８０６号公報

【０００６】

【発明が解決しようとする課題】

ところで、このような従来の複合機では、プリンタ、コピー、スキャナおよびファクシミリ装置に対応するソフトウェアをそれぞれ別個に設けているため、各ソフトウェアの開発に多大の時間を要する。このため、出願人は、表示部、印刷部および撮像部などの画像形成処理で使用されるハードウェア資源を有し、プリンタ、コピーまたはファクシミリなどの各ユーザサービスにそれぞれ固有の処理を行うアプリケーションを複数搭載し、これらのアプリケーションとハードウェア資源との間に介在して、ユーザサービスを提供する際に、アプリケーションの少なくとも２つが共通的に必要とするハードウェア資源の管理、実行制御並びに画像形成処理を行う各種コントロールサービスからなるプラットホームを備えた画像形成装置（複合機）を発明した。

【０００７】

このような新規な複合機では、アプリケーションとコントロールサービスが別個に設けられているため、複合機の出荷後にユーザもしくは第三者であるサードベンダが新規なアプリケーションを開発して複合機に搭載することが可能であり、これによって多種多様な機能を提供することが可能となっている。

【０００８】

このような新規な複合機では、アプリケーションの少なくとも２つが共通的に必要とするサービスを提供するコントロールサービスがアプリケーションと別個に設けられているので、新規アプリケーションを開発する場合、各種コントロールサービスとのプロセス間通信を実現する処理をソースコードで記述する必要がある。

【０００９】

そして、新規アプリケーションを開発する場合は、各コントロールサービスが提供するアプリケーションプログラムインタフェース（ＡＰＩ：関数、イベントを含む）を正確に把握した上で開発しなければならない。しかし、各コントロールサービスのアプリケーションプログラムインタフェース（ＡＰＩ）がデバッグや機能追加によってバージョンアップが繰り返されると、ベンダーはどのバージョンに合わせてアプリケーション開発をすればよいかが非常に分かり難くなり、アプリが使用するＡＰＩのバージョンとコントロールサービスにおけるＡＰＩの

バージョンとが異なる場合も生じ得る。この場合にアプリを実行するとエラーが発生し、複合機のシステムに影響を及ぼす恐れがある。

【００１０】

このことは、固定的な複数の機能を寄せ集めた従来の複合機では問題にならなかった新規な課題である。また、上記コントロールサービス以外にも、アプリケーションにＡＰＩを用いてサービスを提供するモジュールが導入された場合には、同様の問題がある。更に、上記の問題点は画像形成装置について説明したが、この問題点はシステムソフトウェア上でアプリケーションを動作させる一般的な情報処理装置（例えば、一般的なコンピュータ）に共通の問題である。

【００１１】

この発明は上記に鑑みてなされたもので、コントロールサービスなどのシステム側のサービスにおけるＡＰＩのバージョンと、アプリケーションが使用するＡＰＩのバージョンの違いをチェックする技術を提供することを目的とする。

【００１２】

【課題を解決するための手段】

上記目的を達成するため、請求項１に記載の発明は、アプリケーションと、前記アプリケーションからのＡＰＩを用いた要求に基づきシステム側の処理を行うシステムサービスとを備えた情報処理装置であって、前記アプリケーションがシステムサービスに対して使用するＡＰＩのバージョン情報と、当該システムサービスが有するＡＰＩのバージョン情報を取得する取得手段と、前記アプリケーションが前記システムサービスに対して使用するＡＰＩのバージョンを、ＡＰＩ単位に、前記システムサービスが有するＡＰＩのバージョンと比較する比較手段とを有する。

【００１３】

本発明によれば、アプリが使用するＡＰＩ単位毎のバージョン情報とシステムサービスのＡＰＩ単位毎のバージョン情報を取得し、対応するＡＰＩ同士のバージョンが同じか否かをチェックする。ＡＰＩ単位にチェックするので、例えば、アプリが使用していないシステムサービス側のＡＰＩが変更、追加、あるいは削除されて、システムサービスのＡＰＩ全体のバージョンが変わっても、アプリの

動作に影響を与えないことを判断できる。

【００１４】

請求項２に記載の発明は、請求項１の記載において、前記アプリケーションが前記システムサービスに対して使用するＡＰＩのセットのバージョンと、前記システムサービスが有するＡＰＩのセットのバージョンとを比較する手段を更に有し、ＡＰＩのセットのバージョンが異なっている場合にのみ、前記比較手段による比較を行う。

【００１５】

本発明によれば、ＡＰＩのセットのバージョンが異なっている場合にのみＡＰＩ単位の比較を行うので、効率良くバージョンチェックを行うことができる。

【００１６】

請求項３に記載の発明は、請求項１の記載において、前記アプリケーションは、前記システムサービスに対して使用するＡＰＩのバージョン情報を、前記アプリケーションの実行プログラム中に備え、前記取得手段が、当該ＡＰＩのバージョン情報を、前記アプリケーションから取得する取得する。

【００１７】

本発明によれば、ＡＰＩのバージョン情報を、アプリケーションの実行プログラム中に備えるので、アプリケーションからバージョン情報を取得することができる。

【００１８】

請求項４に記載の発明は、請求項３の記載において、前記アプリケーションから前記バージョン情報を取得するために前記アプリケーションを仮起動させるものである。本発明によれば、通常起動をする場合と比較して、情報処理装置のリソースを実質的に確保することがないので効率的にバージョンチェックすることができる。また、通常起動をすることなくバージョンチェックをすることができるので、通常起動することにより情報処理装置に影響を及ぼす恐れがなくなる。

【００１９】

請求項５に記載の発明は、請求項３の記載において、前記システムサービスは複数のシステムサービスを含み、前記アプリケーションは、当該複数のシステム

サービスに対して使用するＡＰＩのバージョン情報を、システムサービス毎に備え、前記取得手段は、前記アプリケーションからあるシステムサービスに対応するＡＰＩのバージョン情報を取得した場合に、当該システムサービスから、当該システムサービスが有するＡＰＩのバージョン情報を取得する。

【００２０】

本発明によれば、アプリが使用する複数のシステムサービスに対してＡＰＩのバージョンチェックを行うことが可能となる。

【００２１】

請求項６に記載の発明は、請求項１の記載において、前記アプリケーションが前記システムサービスに対して使用するＡＰＩのバージョン情報を格納したファイルを備え、前記取得手段が、当該ＡＰＩのバージョン情報を、前記ファイルから取得する。本発明によれば、バージョン情報を記録したファイルを情報処理装置に格納しておくことによりバージョンチェックが可能となる。

【００２２】

請求項７に記載の発明は、請求項１の記載において、前記比較手段による比較を前記アプリケーションのインストール前に行う場合において、前記アプリケーションが前記システムサービスに対して使用する全てのＡＰＩのバージョンが、前記システムサービスが有する対応するＡＰＩのバージョンと一致する場合に、当該アプリケーションがインストール可能であることを表示する。

【００２３】

本発明によれば、バージョンチェックの結果、どのアプリケーションをインストールしてよいかを確認することができる。

【００２４】

請求項８に記載の発明は、請求項１ないし７のうちいずれか１項の記載において、前記情報処理装置は、前記情報処理装置におけるハードウェア資源の制御を行うコントロールサービスと、コントロールサービスをサーバとしたクライアントプロセスとして動作し、前記アプリケーションをクライアントとしたサーバプロセスとして動作する仮想アプリケーションサービスとを有し、前記システムサービスは、アプリケーションからのＡＰＩを用いた要求を受信するコントロール

サービスと、前記仮想アプリケーションサービスを含む。

【００２５】

また、請求項９に記載の発明は、請求項８の記載において、前記仮想アプリケーションサービスは前記取得手段及び前記比較手段を有するものである。

【００２６】

請求項１０～１８に記載の発明は、上記の情報処理装置が実行するバージョンチェック方法の発明であり、請求項１９～２３に記載の発明は、情報処理装置にバージョンチェックの処理を実行させるプログラムの発明である。

【００２７】

請求項２４に記載の発明は、アプリケーションからの要求に基づきシステム側の処理を行うシステムサービスを備えた情報処理装置で実行される当該アプリケーションのプログラムであって、情報処理装置に、前記システムサービスからの要求に基づき、仮起動するか通常起動するかを判断する手順と、仮起動の場合に、前記システムサービスと通信することにより、アプリケーションに関する情報を前記システムサービスに提供する手順とを実行させるプログラムである。

【００２８】

本発明によれば、仮起動をすることができるアプリケーションを提供することが可能となる。

【００２９】

請求項２５に記載の発明は、上記のプログラムを記録したコンピュータ読み取り可能な記録媒体である。

【００３０】

【発明の実施の形態】

以下に添付図面を参照して、この発明の実施の形態にかかる情報処理装置の好適な実施の形態を詳細に説明する。なお、本発明の実施の形態では、情報処理装置の一例として画像形成装置を取り上げて説明する。

【００３１】

（実施の形態１）

図１は、この発明の実施の形態１である画像形成装置（以下、「複合機」とい

う）の構成を示すブロック図である。図１に示すように、複合機１００は、白黒レーザプリンタ（B&W LP）１０１と、カラーレーザプリンタ（Color LP）１０２と、スキャナ、ファクシミリ、ハードディスク、メモリ、ネットワークインタフェースなどのハードウェアリソース１０３を有するとともに、プラットホーム１２０とアプリケーション１３０と仮想アプリケーションサービス（ＶＡＳ：Ｖｉｒｔｕａｌ　Ａｐｐｌｉｃａｔｉｏｎ　Ｓｅｒｖｉｃｅ）１４０から構成されるソフトウェア群１１０とを備えている。

【００３２】

仮想アプリケーションサービス（ＶＡＳ）１４０は、アプリケーション１３０とプラットホーム１２０の間に配置される。ＶＡＳ１４０は、アプリケーション（以下、アプリともいう）１３０の各アプリが初めて登録されるときに、同時に登録処理が行われ、アプリから見るとプラットホーム１２０のサービス層として認識され、サービス層から見るとアプリとして認識されるように登録される。すなわち、ＶＡＳ１４０は、コントロールサービスをサーバとしたクライアントプロセスとして動作し、かつアプリケーションをクライアントとしたサーバプロセスとして動作するものである。

【００３３】

このＶＡＳ１４０はその基本機能としてラッピング機能を備えている。この機能により、アプリ１３０とプラットホーム１２０との間のバージョン差を吸収すると共に、コントロールサービスからのメッセージを取捨選択してプラットホーム１２０を意図的に隠蔽することができる。また、アプリ１３０とコントロールサービスとの間で隠蔽すべきＡＰＩを隠蔽し、特定のＡＰＩだけを開示して重要なＡＰＩの秘匿性を確保することができる。

【００３４】

しかし、バージョン差を吸収するラッピング機能があったとしても、ＶＡＳ１４０についてはコントロールサービス側と整合性がとれるようにバージョンアップが行われるので、アプリケーションとＶＡＳ１４０との間でバージョンの不整合が生じる恐れがあり、その場合、実行時におけるエラーが発生する。

【００３５】

そこで、実行時におけるエラー発生を防止するため、アプリとＶＡＳ１４０の全体バージョン同士を比較して、バージョンが一致しているか否かをチェックする機能を有している。

【００３６】

なお、ここで、ＶＡＳ１４０の全体バージョンとは、ＶＡＳ１４０が提供するＡＰＩのセットのバージョンであり、アプリの全体バージョンとは、アプリがＶＡＳ１４０に対して使用するＡＰＩのセットのバージョンである。

【００３７】

また、アプリがＶＡＳ１４０に対して使用するＡＰＩ毎のバージョンをチェックする機能も有している。この機能を有することにより、アプリとＶＡＳの全体バージョンが違っていても、各ＡＰＩ同士を比較することにより、アプリケーションが動作できると判断した場合には、バージョンの整合性がとれていると判断することができる。

【００３８】

プラットホーム１２０は、アプリケーションからの処理要求を解釈してハードウェア資源の獲得要求を発生させるコントロールサービスと、一または複数のハードウェア資源の管理を行い、コントロールサービスからの獲得要求を調停するシステムリソースマネージャ（ＳＲＭ）１２３と、汎用ＯＳ１２１とを有する。

【００３９】

コントロールサービスは、複数のサービスモジュールから形成され、ＳＣＳ（システムコントロールサービス）１２２と、ＥＣＳ（エンジンコントロールサービス）１２４と、ＭＣＳ（メモリコントロールサービス）１２５と、ＯＣＳ（オペレーションパネルコントロールサービス）１２６と、ＦＣＳ（ファックスコントロールサービス）１２７と、ＮＣＳ（ネットワークコントロールサービス）１２８とから構成される。

【００４０】

また、このプラットホーム１２０における各サービスは、前記アプリケーション１３０からの処理要求を受信し、所定のサービスをアプリケーションに提供するためのアプリケーションプログラムインタフェース（ＡＰＩ）を有している。

なお、本明細書では、上記サービスを提供するための関数の他、アプリケーションからプラットフォーム１２０側に通知されるイベント、プラットフォーム１２０側からアプリケーションに通知されるイベントを含めてＡＰＩと称することとする。

【００４１】

汎用ＯＳ１２１は、ＵＮＩＸ（登録商標）などの汎用オペレーティングシステムであり、プラットホーム１２０並びにアプリケーション１３０の各ソフトウェアをそれぞれプロセスとして並列実行する。

【００４２】

ＳＲＭ１２３のプロセスは、ＳＣＳ１２２とともにシステムの制御およびリソースの管理を行うものである。ＳＲＭ１２３のプロセスは、スキャナ部やプリンタ部などのエンジン、メモリ、ＨＤＤファイル、ホストＩ／Ｏ（セントロＩ／Ｆ、ネットワークＩ／Ｆ、ＩＥＥＥ１３９４　Ｉ／Ｆ、ＲＳ２３２Ｃ　Ｉ／Ｆなど）のハードウェア資源を利用する上位層からの要求にしたがって調停を行い、実行を制御する。

【００４３】

具体的には、このＳＲＭ１２３は、要求されたハードウェア資源の利用が可能であるか（他の要求により利用されていないかどうか）を判断し、利用可能であれば要求されたハードウェア資源が利用可能である旨を上位層に伝える。また、ＳＲＭ１２３は、上位層からの要求に対してハードウェア資源の利用スケジューリングを行い、要求内容（例えば、プリンタエンジンにより紙搬送と作像動作、メモリ確保、ファイル生成など）を直接実施している。

【００４４】

ＳＣＳ１２２のプロセスは、アプリ管理、操作部制御、システム画面表示、ＬＥＤ表示、リソース管理、割り込みアプリ制御などを行う。

【００４５】

ＥＣＳ１２４のプロセスは、白黒レーザプリンタ（B&W LP）１０１、カラーレーザプリンタColor LP）１０２、スキャナ、ファクシミリなどからなるハードウェアリソース１０３のエンジンの制御を行う。

【００４６】

ＭＣＳ１２５のプロセスは、画像メモリの取得および解放、ハードディスク装置（ＨＤＤ）の利用、画像データの圧縮および伸張などを行う。

【００４７】

ＦＣＳ１２７のプロセスは、システムコントローラの各アプリ層からＰＳＴＮ／ＩＳＤＮ網を利用したファクシミリ送受信、ＢＫＭ（バックアップＳＲＡＭ）で管理されている各種ファクシミリデータの登録／引用、ファクシミリ読みとり、ファクシミリ受信印刷、融合送受信を行うためのＡＰＩを提供する。

【００４８】

ＮＣＳ１２８のプロセスは、ネットワークＩ／Ｏを必要とするアプリケーションに対して共通に利用できるサービスを提供するためのプロセスであり、ネットワーク側から各プロトコルによって受信したデータを各アプリケーションに振り分けたり、アプリケーションからデータをネットワーク側に送信する際の仲介を行う。具体的には、ftpd, httpd, lpd, snmpd, telnetd, smtpdなどのサーバデーモンや、同プロトコルのクライアント機能などを有している。

【００４９】

ＯＣＳ１２６のプロセスは、オペレータ（ユーザ）と本体間の情報伝達手段となるオペレーションパネル（操作パネル）の制御を行う。ＯＣＳ１２６は、オペレーションパネルからキー押下をキーイベントとして取得し、取得したキーに対応したキーイベント関数をＳＣＳ１２２に送信するＯＣＳプロセスの部分と、アプリケーション１３０またはコントロールサービスからの要求によりオペレーションパネルに各種画面を描画出力する描画関数やその他オペレーションパネルに対する制御を行う関数などが予め登録されたＯＣＳライブラリの部分とから構成される。このＯＣＳライブラリは、アプリケーション１３０およびコントロールサービスの各モジュールにリンクされて実装されている。なお、ＯＣＳ１２６のすべてをプロセスとして動作させるように構成しても良く、あるいはＯＣＳ１２６のすべてをＯＣＳライブラリとして構成しても良い。

【００５０】

アプリケーション１３０は、ページ記述言語（ＰＤＬ）、ＰＣＬおよびポスト

スクリプト（ＰＳ）を有するプリンタ用のアプリケーションであるプリンタアプリ１１１と、コピー用アプリケーションであるコピーアプリ１１２と、ファクシミリ用アプリケーションであるファックスアプリ１１３と、スキャナ用アプリケーションであるスキャナアプリ１１４と、ネットワークファイル用アプリケーションであるネットファイルアプリ１１５と、工程検査用アプリケーションである工程検査アプリ１１６とを有している。これらの各アプリは、その起動時にＶＡＳ１４０に対して自プロセスのプロセスＩＤとともにアプリ登録要求メッセージを送信し、アプリ登録要求メッセージを受信したＶＡＳ１４０によって、起動したアプリに対する登録が行われるようになっている。

【００５１】

アプリケーション１３０の各プロセス、コントロールサービスの各プロセスは、関数呼び出しとその戻り値送信およびメッセージの送受信によってプロセス間通信を行いながら、コピー、プリンタ、スキャナ、ファクシミリなどの画像形成処理にかかるユーザサービスを実現している。

【００５２】

このように、実施の形態１にかかる複合機１００には、複数のアプリケーション１３０および複数のコントロールサービスが存在し、いずれもプロセスとして動作している。そして、これらの各プロセス内部には、一または複数のスレッドが生成されて、スレッド単位の並列実行が行われる。そして、コントロールサービスがアプリケーション１３０に対し共通サービスを提供しており、このため、これらの多数のプロセスが並列動作、およびスレッドの並列動作を行って互いにプロセス間通信を行って協調動作をしながら、コピー、プリンタ、スキャナ、ファクシミリなどの画像形成処理にかかるユーザサービスを提供するようになっている。また、複合機１００には、サードベンダなどの第三者がコントロールサービス層の上のアプリケーション層に新規アプリ１１７、１１８を開発して搭載することが可能となっている。図１では、この新規アプリ１１７、１１８を搭載した例を示している。

【００５３】

なお、実施の形態１にかかる複合機１００では、複数のアプリケーション１３

０のプロセスと複数のコントロールサービスのプロセスとが動作しているが、アプリケーション１３０とコントロールサービスのプロセスをそれぞれ単一の構成とすることも可能である。また、各アプリケーション１３０は、アプリケーションごとに追加または削除することができる。

【００５４】

図２に複合機１００のハードウェア構成例を示す。

【００５５】

複合機１００は、コントローラ１６０と、オペレーションパネル１７５と、ファックスコントロールユニット（ＦＣＵ）１７６と、プリンタ等の画像形成処理に特有のハードウェア資源であるエンジン部１７７とを含む。コントローラ１６０は、ＣＰＵ１６１と、システムメモリ１６２と、ノースブリッジ（ＮＢ）１６３と、サウスブリッジ（ＳＢ）１６４と、ＡＳＩＣ１６６と、ローカルメモリ１６７と、ＨＤＤ１６８と、ネットワークインターフェースカード（ＮＩＣ）１６９と、ＳＤカード用スロット１７０と、ＵＳＢデバイス１７１と、ＩＥＥＥ１３９４デバイス１７２と、セントロニクス１７３とを含む。なお、メモリ１６２、１６７はＲＡＭ、ＲＯＭ等を含む。ＦＣＵ１７６およびエンジン部１７７は、コントローラ１６０のＡＳＩＣ１６６にＰＣＩバス１７８で接続されている。

【００５６】

ＣＰＵ１６１が、複合機１００にインストールされるアプリケーション、コントロールサービス等のプログラムを、メモリから読み出して実行する。

【００５７】

図３は、実施の形態１にかかる複合機１００のＶＡＳ１４０の構成と、ＶＡＳ１４０と各アプリ、コントロールサービス層１５０および汎用ＯＳ１２１との関係を示すブロック図である。なお、図３では、アプリケーション１３０の例として、プリンタアプリ１１１、コピーアプリ１１２、新規アプリ１１７、１１８を示しているが、他のアプリでも同様の構成である。

【００５８】

仮想アプリケーションサービス（ＶＡＳ）１４０のプロセスには、ディスパッチャ１４５と、制御スレッド１４４と、バージョン情報取得スレッド１４３と、

全体バージョンチェックスレッド１４２と、ＡＰＩ単位バージョンチェックスレッド１４１とが動作している。また、アプリは使用ＡＰＩテーブル２１２を有し、ＶＡＳ１４０は全ＡＰＩテーブル２１１を有している。なお、アプリケーションの起動（仮起動含む）時、ＶＡＳ１４０の起動時には、使用ＡＰＩテーブル２１２、全ＡＰＩテーブル２１１はそれぞれ、例えば、ＲＡＭ２１０に展開される。

【００５９】

ディスパッチャ１４５は、アプリケーション１３０やコントロールサービスからのメッセージ受信を監視し、受信したメッセージに応じて制御スレッド１４４、バージョン情報取得スレッド１４３、全体バージョンチェックスレッド１４２、ＡＰＩ単位バージョンチェックスレッド１４１に処理要求を行うものである。実施の形態１の複合機１００では、ディスパッチャ１４５により、仮起動もしくは通常起動したアプリとＶＡＳ１４０との間で行われるプロセス間通信を行うことができる。なお、仮起動とは、アプリケーションとＶＡＳとが通信をするためだけにアプリケーションを仮に起動することであり、詳細については後述する。

【００６０】

制御スレッド１４４は、ディスパッチャ１４４を介して送られてくる各アプリが使用するＡＰＩ単位毎のバージョン情報や各アプリの全体バージョン情報などを、全体バージョンチェックスレッド１４２、ＡＰＩ単位バージョンチェックスレッド１４１にそれぞれ渡したり、各スレッド１４１～１４３間における処理順序を制御したり、各スレッド１４１～１４３からの処理要求をディスパッチャ１４５に伝えたりする。

【００６１】

また、ＡＰＩ単位バージョンチェックスレッド１４１は、アプリがＶＡＳ１４０に対して使用するＡＰＩ単位毎のバージョン情報と、ＶＡＳ１４０の全ＡＰＩ単位毎のバージョン情報とを取得して、対応するＡＰＩ同士のバージョンを比較して、同じか否かをチェックするものである。

【００６２】

また、全体バージョンチェックスレッド１４２は、各アプリ毎の全体バージョン情報と、ＶＡＳ１４０の全体バージョン情報とを取得して、全体バージョン同士が同じか否かをチェックするものである。このスレッドの機能を有することにより、まず全体バージョン同士を比較して、全体バージョン同士が同じであれば整合性が有りと言えるので、ＡＰＩ単位毎のバージョンチェックを行う必要がなくなり、処理を簡略化することができる。そして、ＡＰＩ単位毎のバージョンチェックは、全体バージョンが違っている場合にのみ行えばよい。

【００６３】

バージョン情報取得スレッド１４３は、ＡＰＩ単位バージョンチェックスレッド１４１で必要なアプリが使用するＡＰＩ単位毎のバージョン情報とＶＡＳのＡＰＩ単位毎のバージョン情報を取得したり、全体バージョンチェックスレッド１４２で必要なアプリの全体バージョン情報とＶＡＳの全体バージョン情報を取得するものである。

【００６４】

本実施の形態では、新規アプリのインストール時にバージョンチェックを行うが、まだアプリが起動されていないためプロセス間通信を使って情報を取得することができない。従って、バージョン情報取得スレッド１４３は、対象となるアプリを仮起動させて、プロセス間通信によりアプリから必要な情報を取得する。なお、アプリから取得する情報として、ＡＰＩ単位毎のバージョン情報以外にもプロダクトＩＤ（ベンダー、アプリ、バージョンから一義的に決定される）、ベンダー名、アプリケーション名、全体バージョン、リソース情報等を取得することも可能である。

【００６５】

なお、図３のように複数のスレッド１４１～１４４を用いる構成とする代わりに、スレッド１４１～１４４を１つのスレッドとして構成してもよい。その場合、その１つのスレッドが、本実施の形態におけるバージョンチェックの一連の処理（バージョン情報取得、バージョンチェック）を実行する。なお、ＶＡＳ１４０のプログラムは、ＳＤカード、ＣＤ－ＲＯＭ等の記録媒体に格納して配布できる。また、ネットワークを介して配布することもできる。また、このプログラム

は、例えば、ＳＤカードを複合機に挿入することにより複合機にインストールすることができる。また、ＳＤカードから起動することもできる。

【００６６】

図４は、本発明のバージョンチェックの方法の概要を説明するための図である。図４に示すように、アプリケーションの実行ファイルに当該アプリケーションがＶＡＳ１４０に対して使用するＡＰＩのバージョン情報（バージョンテーブル）を含めておく。同様に、ＶＡＳ１４０の実行ファイルにも、ＶＡＳが有しているＡＰＩのバージョン情報（バージョンテーブル）を含めておく。なお、一般に、アプリケーションがＶＡＳ１４０に対して使用する複数のＡＰＩは、ＶＡＳ１４０が有している複数のＡＰＩの一部である。

【００６７】

ＶＡＳ１４０が有するＡＰＩのバージョン情報は、ＡＰＩ毎のバージョンと、ＡＰＩのセットとしてのバージョンである。また、アプリケーションが有するバージョン情報は、ＡＰＩ毎のバージョンと、ＡＰＩのセットとしてのバージョンを有している。アプリケーションが使用するＡＰＩセットのバージョンは、アプリケーションを作成するときに使用したＶＡＳのＡＰＩセットのバージョンと同じである。

【００６８】

実行ファイルにバージョン情報を含めるには、例えば、図４に示すように、バージョン情報をインクルードファイルとして作成しておき、作成したプログラムにインクルードしてコンパイルする。

【００６９】

ＶＡＳ１４０は、仮起動されたアプリからプロセス間通信によりアプリが使用するＡＰＩのバージョン情報を取得し、当該バージョン情報と、ＶＡＳが有するＡＰＩのバージョン情報とをＡＰＩ毎に比較して、アプリケーションがインストール可能かどうかを判断する。また、ＡＰＩセットのバージョン同士を比較し、これらが異なる場合にのみＡＰＩ毎のバージョン比較をしてもよい。なお、ＶＡＳ１４０は、アプリが使用するバージョン情報のテーブル全体を取得して比較処理を行ってもよいし、アプリが使用するＡＰＩのバージョン情報をＡＰＩ毎に１

つずつ取得して対応するＡＰＩバージョンとの比較を行ってもよい。

【００７０】

更に、上記のようにバージョン情報を実行ファイルに含める代わりに、アプリのバージョン情報を別ファイルとして複合機１００に格納しておき、ＶＡＳ１４０がそのファイルを参照するようにしてもよい。

【００７１】

バージョン情報はＶＡＳが参照できさえすればよく、バージョン情報を複合機１００内に保持する方法は上記の方法以外の種々の方法を用いることができる。

【００７２】

さて、アプリが作成された後に、ＶＡＳ１４０のＡＰＩセットのバージョンが上がり、アプリが使用するＡＰＩセットのバージョンと、ＶＡＳが有するＡＰＩセットのバージョンとが異なる場合が起こり得る。

【００７３】

例えば、図５の例では、ＡＰＩ　Ｎｏ．＝２５１のバージョンが（１０１）→（１０２）にバージョンアップしたために、ＶＡＳ１４０のＡＰＩセットのバージョンが（１．００）→（１．０１）となっている。

【００７４】

この場合、ＡＰＩセットを比較しただけでは、バージョンが異なるので、アプリとＶＡＳとの整合性がないと判断され得るが、実際には、ＡＰＩ　Ｎｏ．２５１はアプリで使用しないＡＰＩであるため、アプリは問題なく動作できる。従って、ＡＰＩ毎の比較を行うことにより、ＡＰＩセット同士のバージョンが異なっていても、アプリが問題なく動作できると判断できる。

【００７５】

次に、アプリケーションの仮起動について説明する。仮起動は、複合機のリソースを使用することになるアプリの通常起動（アプリ本来の機能を奏するための起動を通常起動と呼ぶ）とは別の起動である。仮起動では、アプリはアプリ本来の動作に必要なメモリ確保等のリソース取得を行わず、ＶＡＳ１４０とのプロセス間通信処理のみを行う。そして、アプリは、ＶＡＳ１４０がバージョンチェックを含むアプリに関するチェックを行うために必要な情報をＶＡＳ１４０に提供

する。仮起動したアプリのプロセスは、ＶＡＳ１４０との通信処理が終了すれば終了する。また、アプリの仮起動に関する機能は、アプリ本来の機能によらず、本実施の形態における複合機１００で動作するアプリに共通する機能である。従って、例えば、ベンダーがアプリを開発する場合、ベンダーに、仮起動の機能を含むプログラムテンプレートを提供し、そのプログラムテンプレートを用いてベンダーが複合機用のアプリを開発することができる。なお、上記のバージョンチェックを実現するためには、例えば、ベンダーが、アプリ開発に使用したＡＰＩとそのバージョンを記録したインクルードファイルを作成し、アプリのコンパイル時にインクルードする。

【００７６】

アプリが仮起動の機能を持つことにより、通常起動をすることなくＶＡＳとの通信によりＶＡＳにアプリ情報を提供でき、ＶＡＳがアプリのチェックを行うことができる。従って、ＡＰＩバージョン等の整合性がとれていないアプリを通常起動することによりアプリが異常動作し、複合機に影響を与える恐れがなくなる。

【００７７】

図６に、仮起動の機能を含むアプリのプログラム記述（メイン関数）の概要を示す。なお、この記述を上記のプログラムテンプレートとして提供する。

【００７８】

図６に示すように、このプログラム記述は、アプリケーションを仮起動するか通常起動するかを引数（－ｖ）によって指定する。これにより、ＶＡＳがアプリを起動する際に、通常起動と仮起動とを容易に使い分けることができる。すなわち、引数（－ｖ）を使って仮起動を指定すると、仮起動が実行され、アプリ情報提供処理がなされる。また、仮起動が指定されていない場合は、通常起動を行って、アプリ本来の動作を行う。

【００７９】

なお、後述するアプリ起動時、あるいは、アプリ起動後の実行時にバージョンチェックする場合は、この通常起動処理が選択される。

【００８０】

このように仮起動の機能を有するアプリのプログラムは、ＳＤカード、ＣＤ－ＲＯＭ等の記録媒体に格納して配布できる。また、ネットワークを介して配布することもできる。また、このプログラムは、例えば、ＳＤカードを複合機に挿入することにより複合機にインストールすることができる。また、ＳＤカードから起動することもできる。

【００８１】

次に、アプリインストール時にバージョンチェックを行う場合の複合機の動作について、図７のフローチャートを用いて説明する。図８は、図７の中のバージョンチェック処理のサブルーチンを示すフローチャートである。

【００８２】

まず、図７に示すように、ユーザがオペレーションパネル上のキーやボタンを使ってアプリのインストール処理を選択すると（ステップＳ５０１）、ＳＣＳ（システムコントロールサービス）１２２からＶＡＳ１４０にアプリインストール処理の開始を要求し（ステップＳ５０２）、ＶＡＳ１４０がインストール対象エリア内のアプリを仮起動させる（ステップＳ５０３）。この仮起動は、前述したように、インストールするアプリ内の各種情報をプロセス間通信を使ってＶＡＳ１４０に通知し、ＶＡＳがバージョンチェック処理を行うためのものである（ステップＳ５０４）。

【００８３】

ステップＳ５０４におけるバージョンチェック処理の詳細を図８を用いて説明する。

【００８４】

図８に示すように、アプリから通知された情報にアプリの全体バージョン情報（ＡＰＩセットのバージョン情報）が含まれているか否かを判断し（ステップＳ６０１）、全体バージョン情報が含まれている場合は、ＶＡＳ１４０のＡＰＩの全体バージョンと比較し、同じであれば（ステップＳ６０２）、バージョンの不整合はなく、ＶＡＳ１４０は当該アプリケーションに対して動作保証する（ステップＳ６０３）。なお、動作保証するとは、例えば、当該アプリに対応づけてＡＰＩバージョンに関して問題ない旨のフラグを記録しておくことである。

【００８５】

また、図８のステップＳ６０１もしくはステップＳ６０２おいて、ＮＯであった場合には、ステップＳ６０４に移行し、アプリが使用するＡＰＩのバージョンと、それに対応したＶＡＳ１４０のＡＰＩのバージョンとが、アプリが使用するＡＰＩ毎に１つ１つ比較され、全て同じバージョンであるか否かが判断され、同じであれば上記ステップＳ６０３にてＶＡＳ１４０のサポート範囲内として動作保証する（ステップＳ６０３）。しかしステップＳ６０４でバージョンの異なるＡＰＩがあった場合は、ＶＡＳ１４０のサポート範囲外となるため、その旨を記録する（ステップＳ６０５）。

【００８６】

このように、図８のステップＳ５０４のサブルーチンでバージョンチェック処理を行った後、他にインストール処理を行うアプリが無くなるまで上記ステップＳ５０３に戻り、上記バージョンチェック処理が繰り返される（ステップＳ５０５）。

【００８７】

インストール処理するアプリが無くなった場合は、ステップＳ５０６でインストール時に取得したアプリの情報に基づいてインストール画面をオペレーションパネル上などに生成し、各種情報を表示する（ステップＳ５０６）。例えば、インストール処理された複数のアプリのバージョンチェック結果に基づいて動作保証の有無を表示する。これにより、ユーザは、正式なインストール要求すべきアプリをインストール画面から容易に選択することができ（ステップＳ５０７）、選択されたアプリのみを最終的にインストール処理する（ステップ５０８）。

【００８８】

上記したように、実施の形態１では、アプリのインストール時に、ＶＡＳによってアプリを仮起動させ、プロセス間通信を利用してバージョン情報を含むアプリに関する情報を取得して、バージョンチェック行うことができるので、動作保証されたアプリのみを選択的にインストール処理することができる。

【００８９】

また、インストール時の他、アプリの通常起動時に本発明のバージョンチェッ

ク処理を行うことができる。アプリを仮起動させるか通常起動させるかは、図６のメイン関数記述例に示したように、引数（－ｖ）を用いるか否かで決まり、通常起動時にも勿論プロセス間通信を使ってアプリのバージョン情報などが取得できるため、バージョンチェック処理を行うことができる。このアプリの起動時におけるバージョンチェックは、既にインストールされているアプリを初めて起動する場合などに有効である。また、アプリのバージョン情報を別ファイルに持たせておけばプロセス間通信を行わず、ＶＡＳが当該ファイルを参照することによりチェックを行うことができる。

【００９０】

さらに、アプリ起動後の実行時にもバージョンチェック処理を行うことができる。アプリ起動後の実行時には、プロセス間通信が使えるため、アプリの通常起動時におけるバージョンチェックと同じように適切なタイミングでバージョンチェックを行うことができる。

【００９１】

また、アプリの実行時におけるバージョンチェックとして、例えば、アプリ側でＡＰＩの引数の中にバージョン情報を入れておき、そのＡＰＩをアプリが使用するときにＶＡＳ１４０にＡＰＩのバージョンを通知し、ＶＡＳ１４０がそのバージョンをチェックするようにしてもよい。ＶＡＳ１４０が、ＡＰＩのバージョン差があると判断したときは、ＶＡＳ１４０がそのＡＰＩの要求をサービス層まで伝えないようにする。更に、アプリの実行処理を中止し、オペレーションパネル上にエラー表示を行って、ユーザにその旨を通知するようにしてもよい。バージョン差がない場合には、アプリの実行がそのまま継続して行われる。

【００９２】

なお、バージョンチェック処理をＶＡＳ１４０側でなく、アプリ側で行ってもよい。その場合、アプリがＶＡＳ１４０からＶＡＳ１４０の有するＡＰＩのバージョン情報を取得し、バージョンチェック処理を行う。そして、インストール時であればバージョン差があるか否かをＶＡＳ１４０に伝える。また、起動時、実行時であれば、バージョン差がなければ起動もしくは実行を継続し、バージョン差があれば、例えば、その旨のオペレーションパネル上に表示し、処理を終了す

る。

【００９３】

このように、実施の形態１にかかる複合機では、ＶＡＳ１４０が、アプリが使用するＡＰＩ単位毎のバージョン情報とＶＡＳのＡＰＩ単位毎のバージョン情報を取得し、対応するＡＰＩ同士のバージョンが同じか否かをチェックする。従って、アプリが使用していないＶＡＳ１４０のＡＰＩが変更、追加、あるいは削除されて、ＶＡＳのＡＰＩ全体（ＡＰＩセット）のバージョンが変わっても、両者間の整合性に影響を与えないことを判断できる。全体バージョン同士を単純比較する場合と比べると、ＶＡＳのバージョン差の吸収範囲が広くなって、利用可能なアプリを増やすことができる。

【００９４】

また、実施の形態１にかかる複合機では、ＶＡＳ１４０が最初に全体バージョンを比較し、全体バージョンが異なっている場合のみＡＰＩ単位のバージョンチェックをすることができるので、バージョンチェック処理を効率良く、かつ迅速に行うことができる。また、予め必要なバージョン情報を持たせたアプリの実行ファイルを使うことにより、アプリの仮起動や通常起動の際のプロセス間通信により、ＶＡＳが所望のバージョン情報を取得することができる。

【００９５】

また、アプリのインストール時にバージョンチェックする場合、ＶＡＳ１４０がアプリを仮起動させて、アプリが使用するＡＰＩ単位毎のバージョン情報を得ることができるので、通常起動をする場合と比較して効率的にバージョンチェックすることができる。

【００９６】

（実施の形態２）

次に、実施の形態２について説明する。実施の形態１では、図４に示すように、アプリはアプリがＶＡＳ１４０に対して使用するＡＰＩのバージョン情報を有し、ＶＡＳ１４０はそのバージョン情報をチェックしていた。

【００９７】

さて、複合機におけるＶＡＳ１４０は、図１に示すように全てのコントロール

サービスをカバーするのではなく、一部のコントロールサービスのみをカバーするように構成することもできる。実施の形態２では、ＶＡＳ１４０を介さずに直接アプリと通信を行うコントロールサービスに対するＡＰＩのバージョンもチェックできる構成について説明する。なお、以下の説明において、ＶＡＳ、コントロールサービスを含めてそれぞれ“システム”と呼ぶ。

【００９８】

実施の形態２では、図９に示すように、アプリは、ＶＡＳ１４０に対して使用するＡＰＩのバージョン情報の他、ＶＡＳ１４０を介さずに通信を行うコントロールサービスに対して使用するＡＰＩのバージョン情報を有している。ＶＡＳ１４０は、アプリからこれらのバージョン情報を取得し、該当するシステムからそのシステムのＡＰＩのバージョン情報を取得し、バージョン比較を行う。

【００９９】

図９に示すように、アプリは、各システム毎にバージョン情報のテーブルを有していてもよいし、アプリが使用する全システムについてのバージョン情報を１つのテーブルに保持していてもよい。この場合、テーブル内でどのシステムのＡＰＩかは識別できる。図９の場合、各テーブルは、ＡＰＩ毎のバージョンに加えてＡＰＩセットのバージョンも保持している。なお、アプリは、図１０に示すような、システム毎のＡＰＩセットのバージョンを保持したテーブルを持っていてもよい。

【０１００】

なお、図９に示す各バージョン情報は、アプリなどのプログラムの実行ファイル内に保持する他、別ファイルに備えておいてもよい。その場合、ＶＡＳ１４０はプロセス間通信でなく、そのファイルを開くことによりバージョン情報を取得する。バージョン情報を保持するにはその他にも種々の方法が可能である。

【０１０１】

図１１に、図９の構成における、ＶＡＳによるバージョンチェック処理のフローチャートを示す。なお、図１１のフローは、ＶＡＳが、アプリからバージョン情報のテーブルを順次取得することを前提としている。また、インストール時のバージョンチェック処理を示している。

【０１０２】

まず、ＶＡＳがアプリから１つのテーブルを取得する（ステップＳ７０１）。そして、そのテーブルがどのシステムに対応するものであるかを判断する（ステップＳ７０２）。次に、該当するシステムが有するバージョン情報のテーブルをそのシステムから取得する（ステップＳ７０３）。

【０１０３】

次に、アプリから取得したテーブルと、システムから取得したテーブルとを比較し、全バージョン（ＡＰＩセットのバージョン）が一致するか否かを判断する（ステップＳ７０４）。一致する場合には、そのシステムに対してはバージョンが整合している旨を記録する（ステップＳ７０６）。そして、アプリ側から取得すべきテーブルが他に無ければバージョンチェック処理を終了する。他にあればステップＳ７０１からの処理を繰り返す。

【０１０４】

ステップＳ７０４にて、一致していない場合には、アプリがそのシステムに対して使用するＡＰＩ毎にバージョンをチェックする（ステップＳ７０５）。全ＡＰＩのバージョンが同じであれば、そのシステムに対してはバージョンが整合している旨を記録する（ステップＳ７０６）。また、１つでもバージョンの異なるＡＰＩがある場合には、バージョンが整合していない旨を記録する。このとき、どのＡＰＩのバージョンが整合していないかを記録しておいてもよい。そして、アプリ側から取得すべきテーブルが他に無ければバージョンチェック処理を終了する。他にあればステップＳ７０１からの処理を繰り返す。

【０１０５】

バージョンチェック処理が終了した後、ＶＡＳ１４０は、例えば、そのアプリが使用する全てのシステムに対してバージョンが一致していれば、そのアプリがインストール可能であることをオペレーションパネルに表示し、１つでもバージョンの不一致があるシステムがあれば、バージョンに不一致がある旨を表示する。バージョンに不一致がある旨を表示する際、どのシステムのどのＡＰＩに不一致があるかを表示してもよい。

【０１０６】

本実施の形態により、ＶＡＳ１４０に対するＡＰＩのみならず、ＶＡＳ１４０を介さずに通信を行うＡＰＩのバージョンもチェックできる。

【０１０７】

（実施の形態３）

実施の形態１にかかる複合機１００は、ＶＡＳ１４０が全アプリケーションに対して１つのみ存在するものであった。この実施の形態２にかかる複合機９００では、各アプリごとに一つのＶＡＳを起動し、各ＶＡＳは対応するアプリとの間でバージョンチェック処理を行うようにする。

【０１０８】

図１２は、実施の形態３にかかる複合機９００の構成を示すブロック図である。図１２に示すように、複合機９００では、複数の仮想アプリケーションサービス（ＶＡＳ）９４１～９４８がアプリケーション１３０の各アプリごとに動作している点が、実施の形態１にかかる複合機１００と異なっている。

【０１０９】

ＶＡＳ９４１～９４８は、プリンタアプリ１１１、コピーアプリ１１２、ファックスアプリ１１３、スキャナアプリ１１４、ネットファイルアプリ１１５、工程検査アプリ１１６、新規アプリ１１７および１１８に対応して、バージョンチェック処理を行う。

【０１１０】

図１３は、実施の形態３にかかる複合機９００のＶＡＳ９４１～９４８の構成と、ＶＡＳ９４１～９４８と各アプリ、コントロールサービス層１５０および汎用ＯＳ１２１との関係を示すブロック図である。なお、図１３では、アプリケーション１３０として、プリンタアプリ１１１、コピーアプリ１１２、新規アプリ１１７、１１８の例を示し、さらにこれら各アプリに対応したＶＡＳ９４１、９４２、９４７および９４８を例として示しているが、他のアプリを使用する場合も同様の構成である。

【０１１１】

また、実施の形態３にかかる複合機９００では、実施の形態１の複合機１００と異なり、図１３に示すように、各ＶＡＳ９４１～９４８と各アプリとの間には

ＶＡＳ制御プロセス（デーモン）９０１が動作している。このＶＡＳ制御プロセス（デーモン）９０１は、ＶＡＳが提供する全ＡＰＩのバージョン情報を有している。

【０１１２】

仮想アプリケーションサービス（ＶＡＳ）９４１～９４８のプロセスは、ディスパッチャ１４５と、ＡＰＩ単位バージョンチェックスレッド１４１と、全体バージョンチェックスレッド１４２と、バージョン情報取得スレッド１４３とが動作している。

【０１１３】

実施の形態３の複合機９００における各スレッドの機能は、実施の形態１で説明したものと同様である。なお、スレッド１４１～１４３を１つのスレッドとしてもよい。

【０１１４】

また、ＶＡＳの構成として上記の構成の他、図１４（ａ）～（ｃ）に示す構成をとることもできる。

【０１１５】

図１４（ａ）は、各アプリケーションに対して起動されるＶＡＳを、親ＶＡＳの子プロセスとする場合であり、親ＶＡＳ自体は画面制御権（ユーザインターフェース）を持たない。図１４（ｂ）は、親ＶＡＳ自体が画面制御権（ユーザインターフェース）を持つ場合である。図１４（ｃ）は、各アプリケーションに対応するＶＡＳの機能をスレッドとして起動する場合を示している。

【０１１６】

例えば、図１４（ｃ）の場合において、アプリが本起動していない時点では、アプリ対応のＶＡＳスレッドは起動されていない。アプリのインストール時に、アプリと対応付けられない１つのＶＡＳがアプリの仮起動を行い、バージョンチェックを行う。その後、アプリの本起動時に各アプリに対応したＶＡＳのスレッドが起動される。

【０１１７】

なお、本発明は、上記の実施の形態に限定されることなく、特許請求の範囲内

において、種々変更・応用が可能である。

【０１１８】

【発明の効果】

上記のように、本発明によれば、アプリが使用するＡＰＩ単位毎のバージョン情報とＶＡＳ等のシステムサービス側のＡＰＩ単位毎のバージョン情報を比較し、対応するＡＰＩ同士のバージョンが同じか否かをチェックできるので、アプリを正常に使用できるか否かを判断することができる。

【図面の簡単な説明】

【図１】

本発明の実施の形態１における複合機のソフトウェア構成を示すブロック図である。

【図２】

本発明の実施の形態１における複合機のハードウェア構成を示すブロック図である。

【図３】

本発明の実施の形態１における複合機のＶＡＳの構成と各アプリとコントロールサービス層と汎用ＯＳとの関係を示すブロック図である。

【図４】

本発明の実施の形態１におけるバージョンチェック方法を説明するための図である。

【図５】

本発明の実施の形態１におけるバージョンチェック方法を説明するための図である。

【図６】

本発明の実施の形態１における仮起動可能なアプリのプログラム記述例である。

【図７】

アプリインストール時におけるバージョンチェックのシーケンスを説明するためのフローチャートである。

【図８】

図７の中のバージョンチェック処理を示すフローチャートである。

【図９】

本発明の実施の形態２におけるバージョンチェックの方法を説明するための図である。

【図１０】

本発明の実施の形態２においてアプリが有するテーブルの例である。

【図１１】

本発明の実施の形態２におけるバージョンチェック処理の手順を示すフローチャートである。

【図１２】

本発明の実施の形態３における複合機のソフトウェア構成を示すブロック図である。

【図１３】

本発明の実施の形態３における複合機のＶＡＳの構成と、ＶＡＳと各アプリ、コントロールサービス層および汎用ＯＳとの関係を示すブロック図である。

【図１４】

本発明の実施の形態におけるＶＡＳの構成例を示す図である。

【符号の説明】

１００　複合機

１０１　白黒レーザプリンタ

１０２　カラーレーザプリンタ

１０３　ハードウェアリソース

１１０　ソフトウェア群

１１１　プリンタアプリ

１１２　コピーアプリ

１１３　ファックスアプリ

１１４　スキャナアプリ

１１５　ネットファイルアプリ

１１６　工程検査アプリ

１１７、１１８　新規アプリ

１２０　プラットホーム

１２１　汎用ＯＳ

１２２　ＳＣＳ

１２３　ＳＲＭ

１２４　ＥＣＳ

１２５　ＭＣＳ

１２６　ＯＣＳ

１２７　ＦＣＳ

１２８　ＮＣＳ

１３０　アプリケーション

１４０、９４１～９４８　仮想アプリケーションサービス（ＶＡＳ）　１４１　ＡＰＩ単位バージョンチェックスレッド

１４２　全体バージョンチェックスレッド

１４３　バージョン情報取得スレッド

１４４　制御スレッド

１４５　ディスパッチャ

１５０　コントロールサービス層

２０１　ラッピング処理情報ファイル

２１０　ＲＡＭ

２１１　全ＡＰＩテーブル

２１２　使用ＡＰＩテーブル

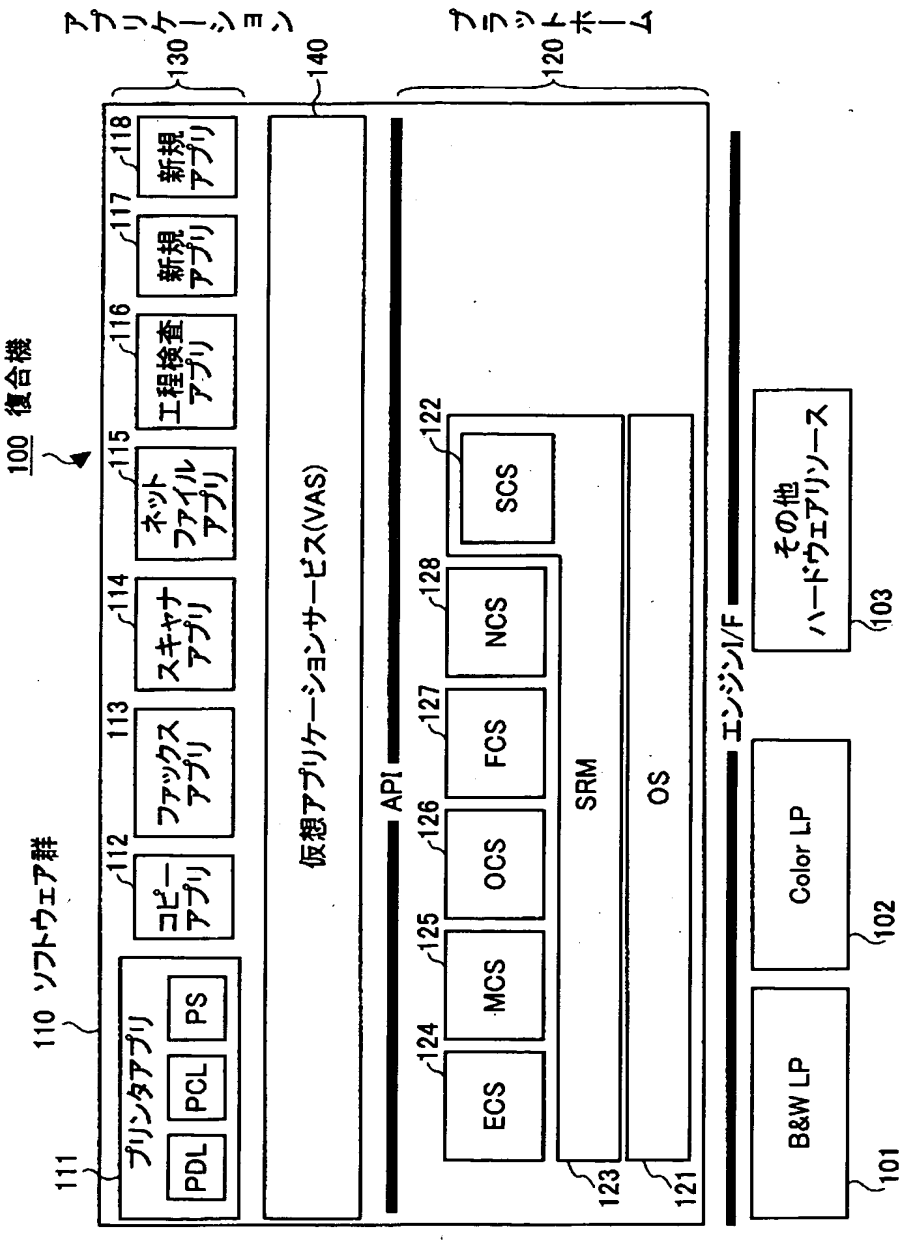
９００　複合機

９０１　ＶＡＳ制御プロセス

【書類名】　図面

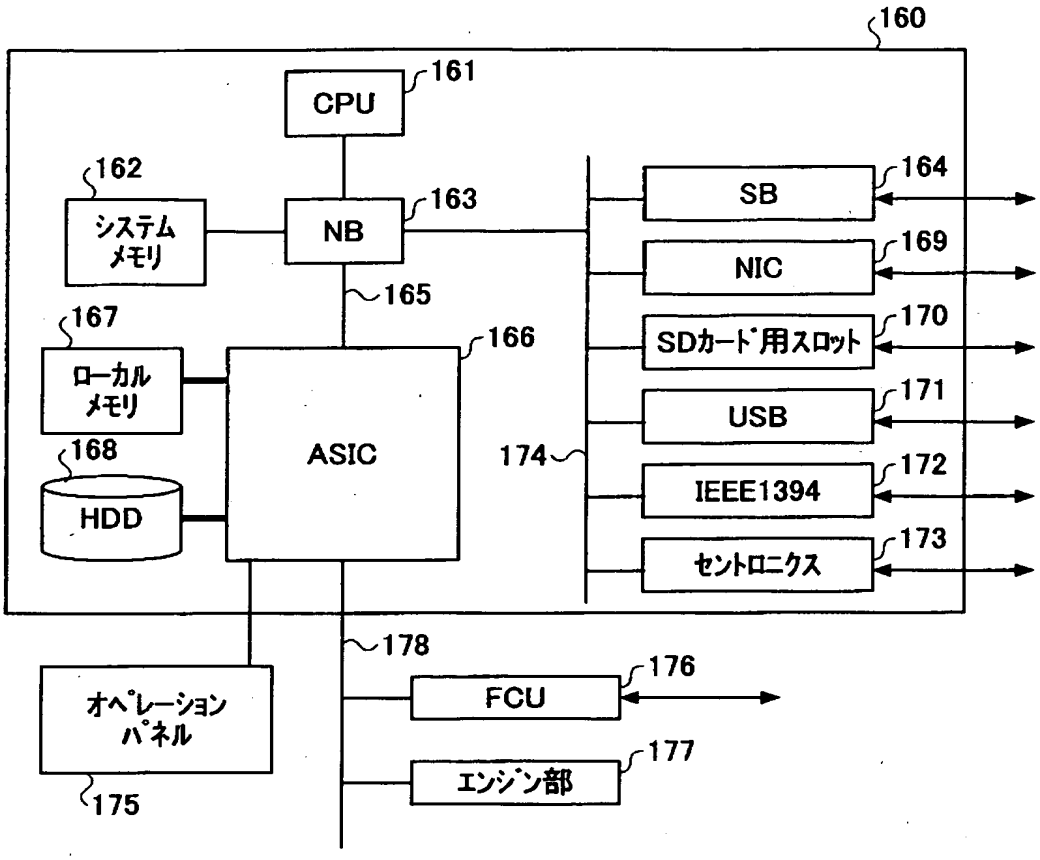
【図１】

本発明の実施の形態1における複合機のソフトウェア構成を示すブロック図



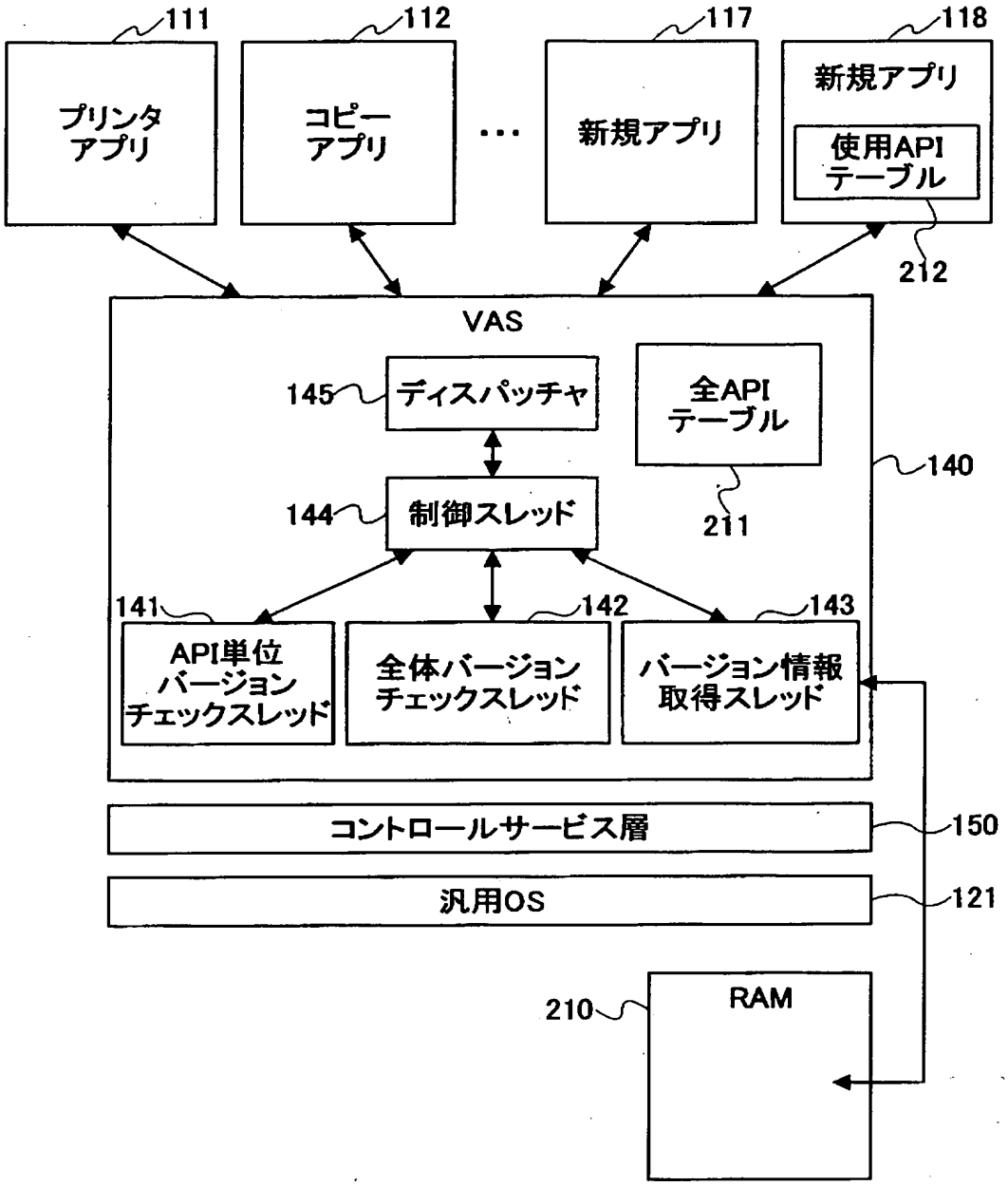
【図２】

本発明の実施の形態1における複合機の
ハードウェア構成を示すブロック図



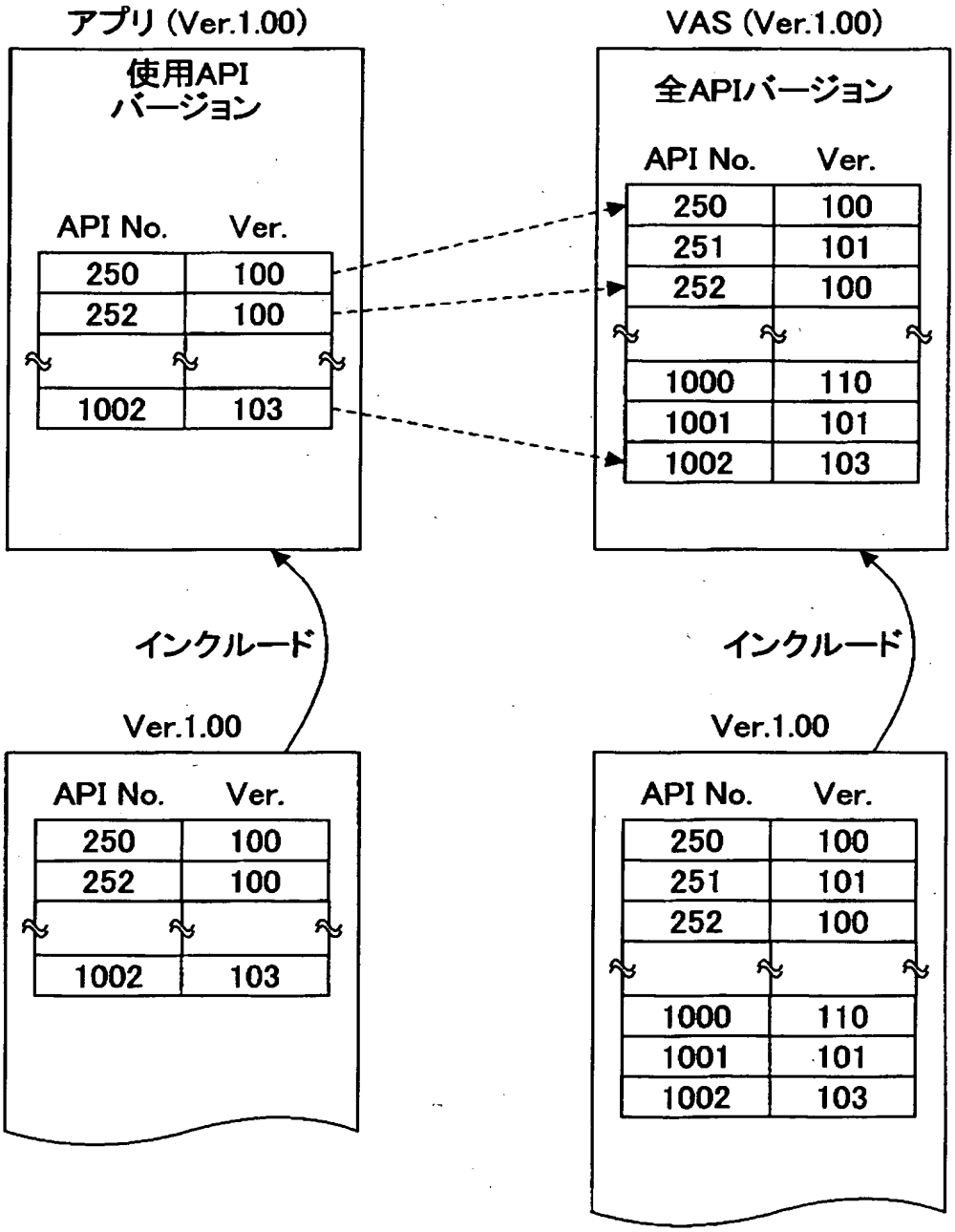
【図３】

本発明の実施の形態1における複合機のVASの構成と
各アプリとコントロールサービス層と
汎用OSとの関係を示すブロック図



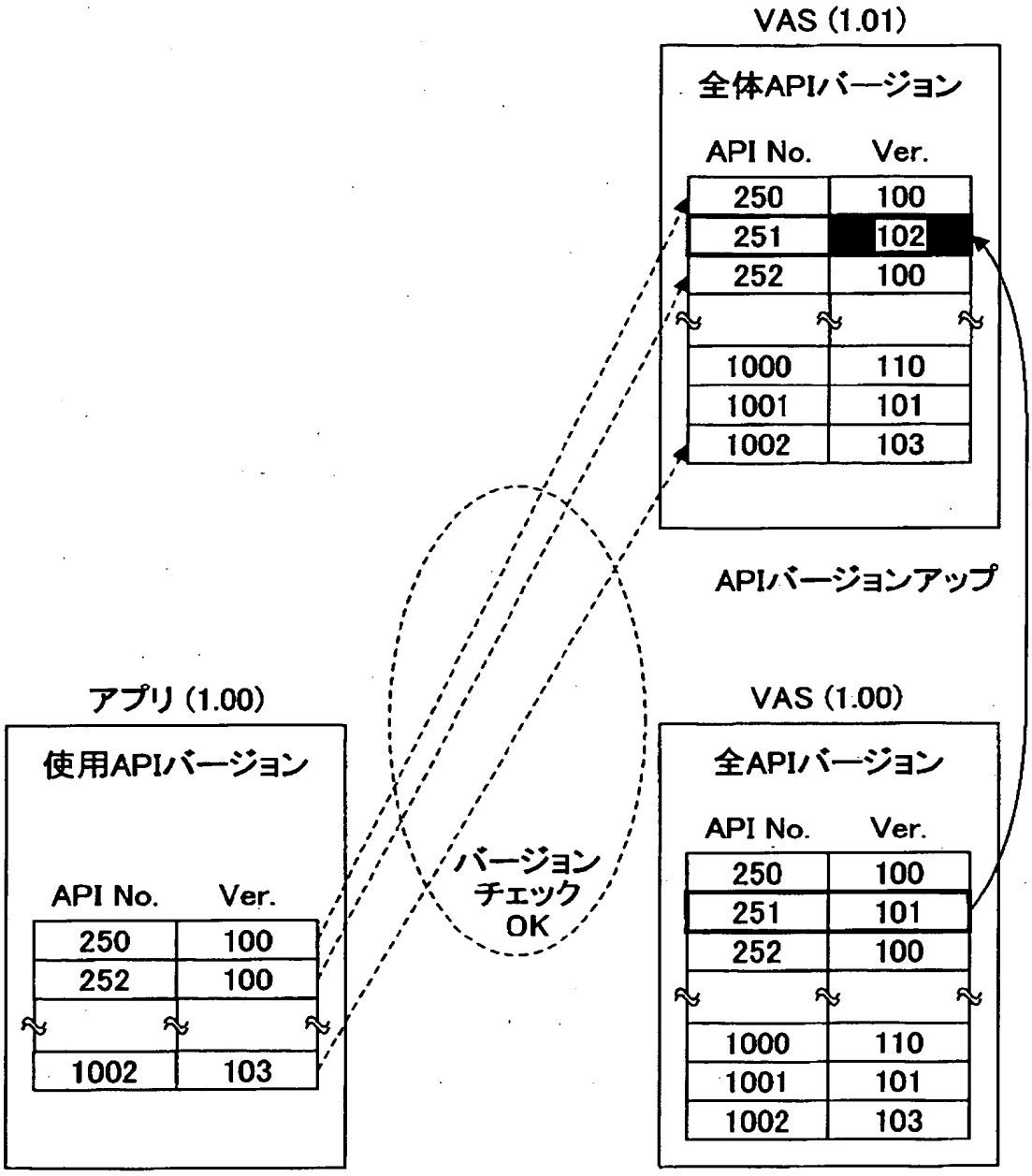
【図４】

本発明の実施の形態1におけるバージョンチェック方法を説明するための図



【図５】

本発明の実施の形態1におけるバージョンチェック方法を説明するための図



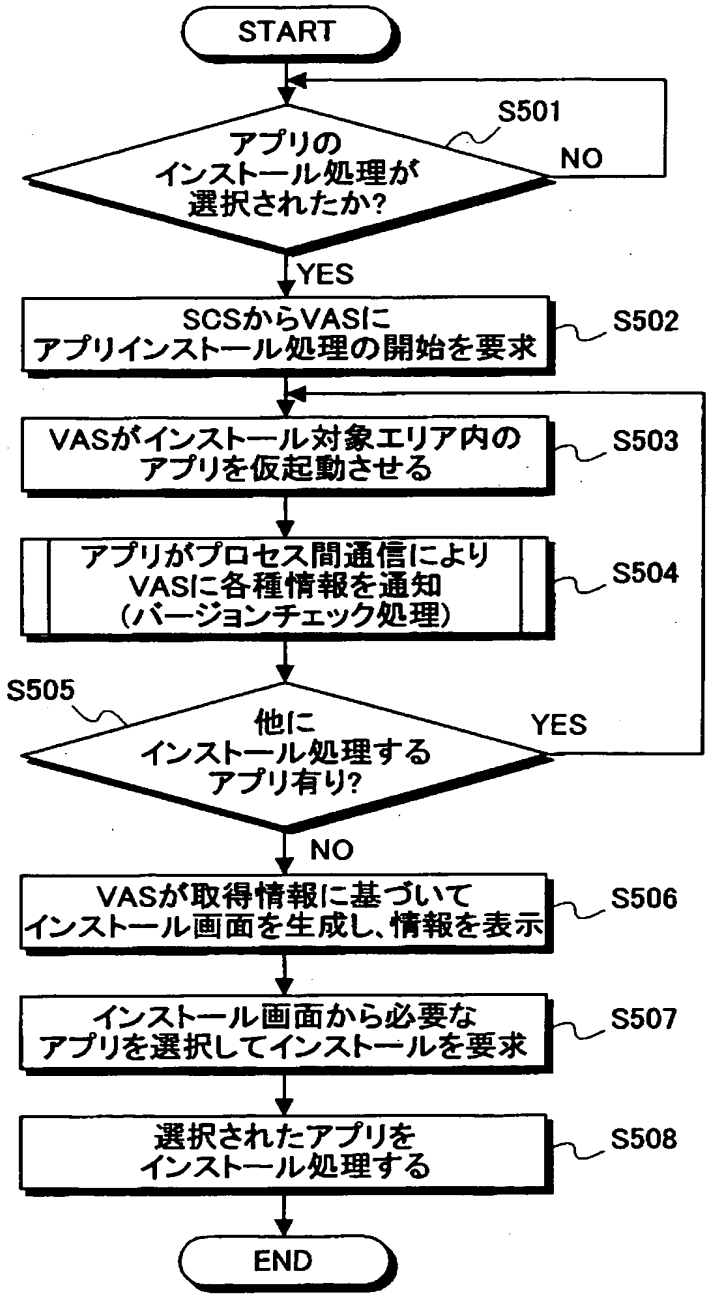
【図６】

本発明の実施の形態1における
仮起動可能なアプリのプログラム記述例

```
int main(int argc, char**argv)
{
  if(argc==2) {
      if(strcmp(argv[1], "-v")==0)
            //仮起動実行
             if(ConnectVAS()==OK){  //VASとの通信準備
                  SetApliInfo();        //アプリ情報提供処理
                       Close            //通信終了
              }
              exit(1)
       }
   }
  通常起動処理
}
```

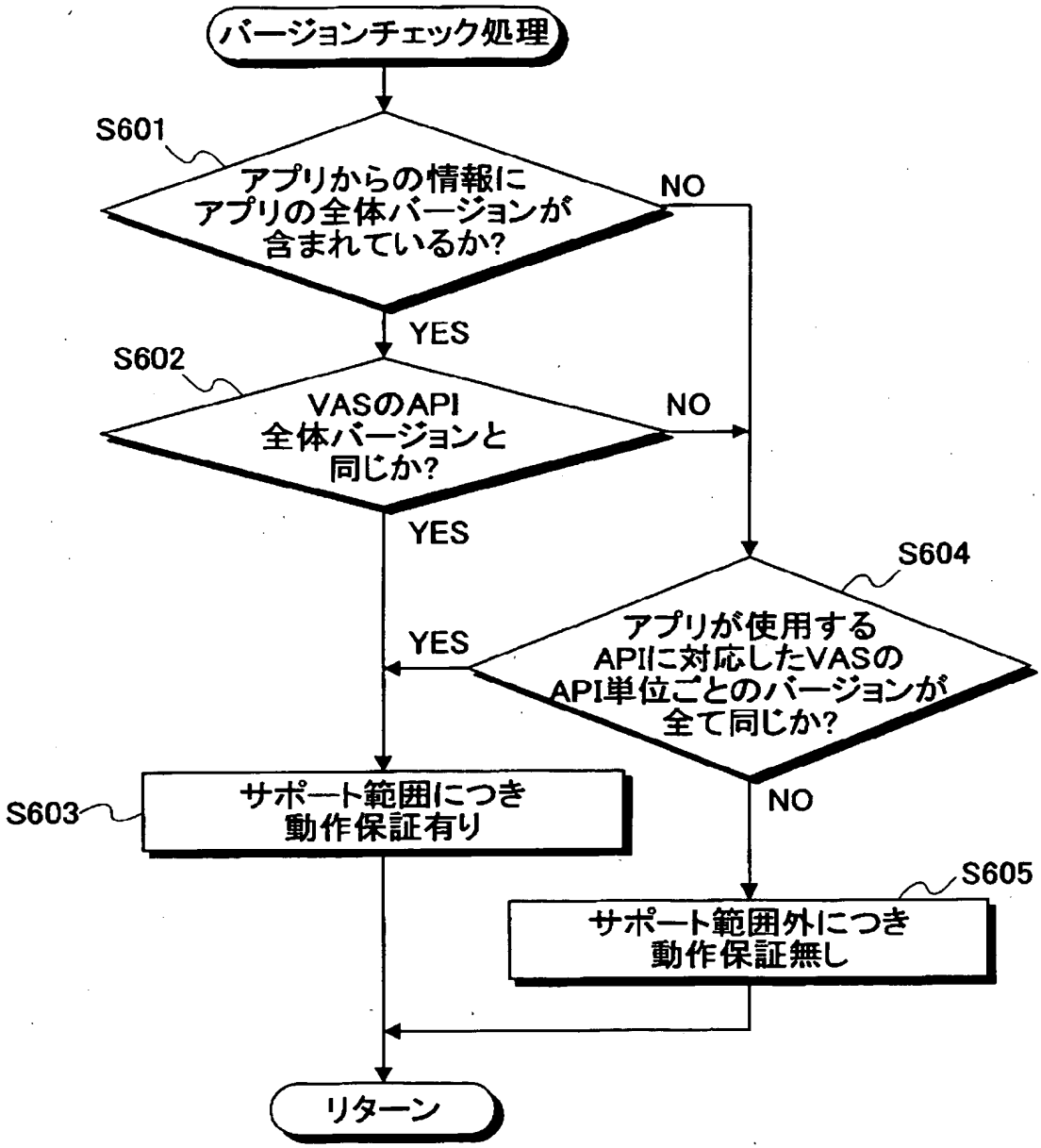
【図７】

アプリインストール時におけるバージョンチェックのシーケンスを説明するためのフローチャート



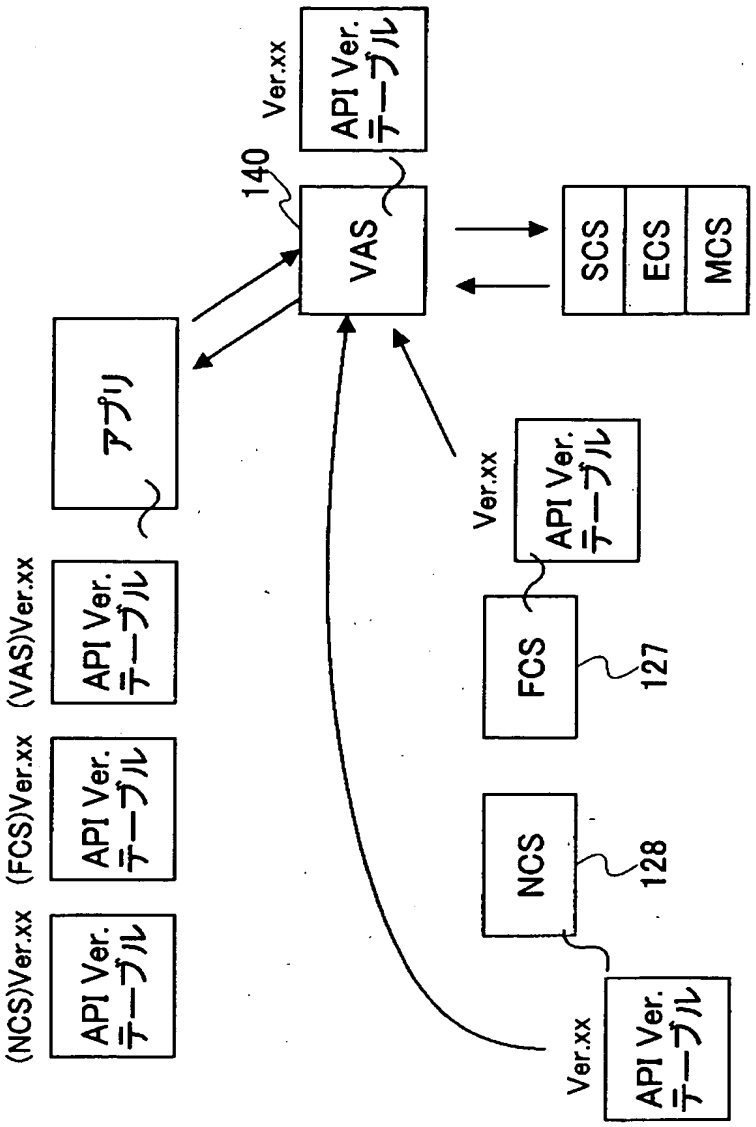
【図８】

図7の中のバージョンチェック処理を示すフローチャート



【図９】

本発明の実施の形態2におけるバージョンチェックの方法を説明するための図



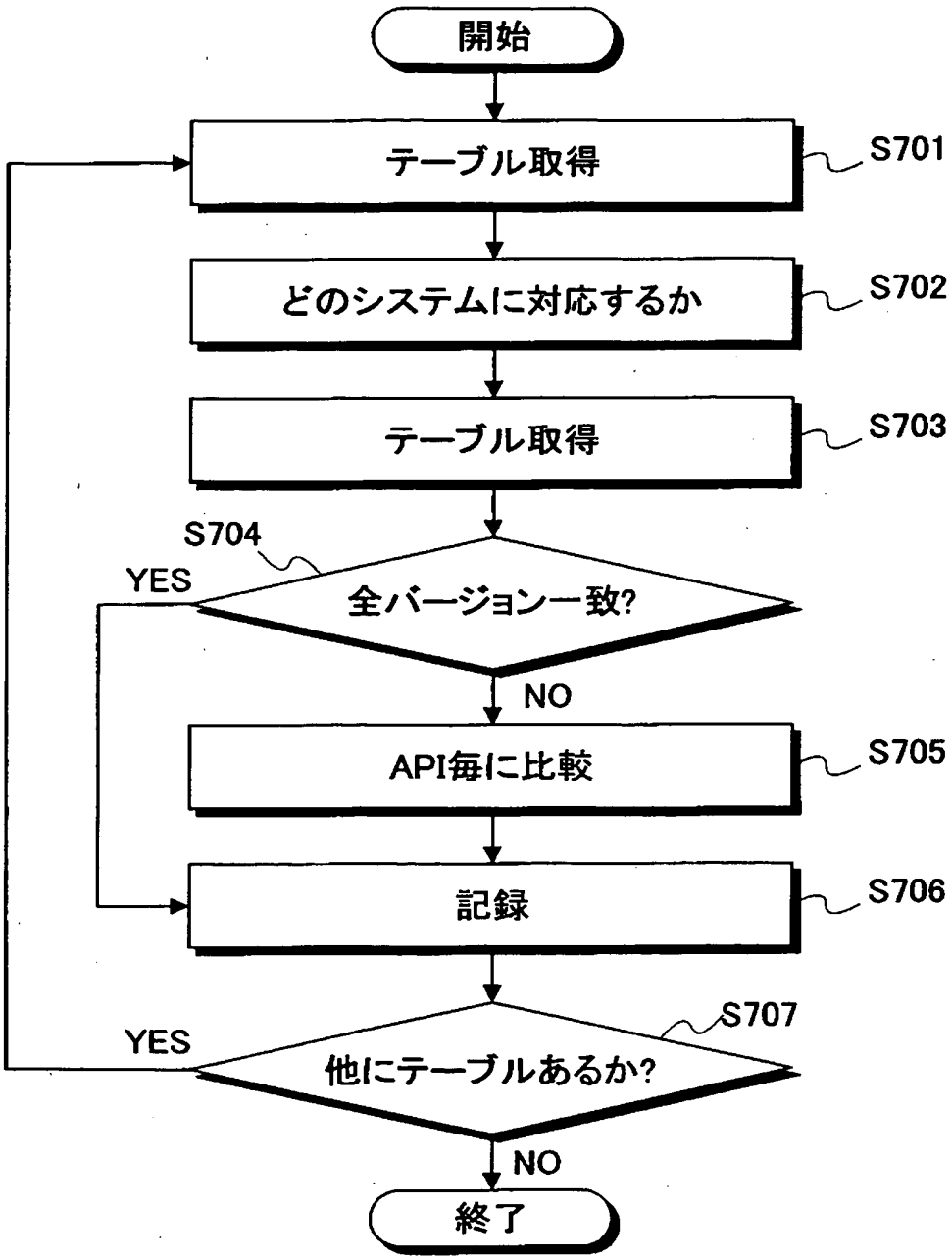
【図１０】

本発明の実施の形態2においてアプリが有するテーブルの例

NCS	Ver. 1.2
FCS	Ver. 2.5
VAS	Ver. 3.0

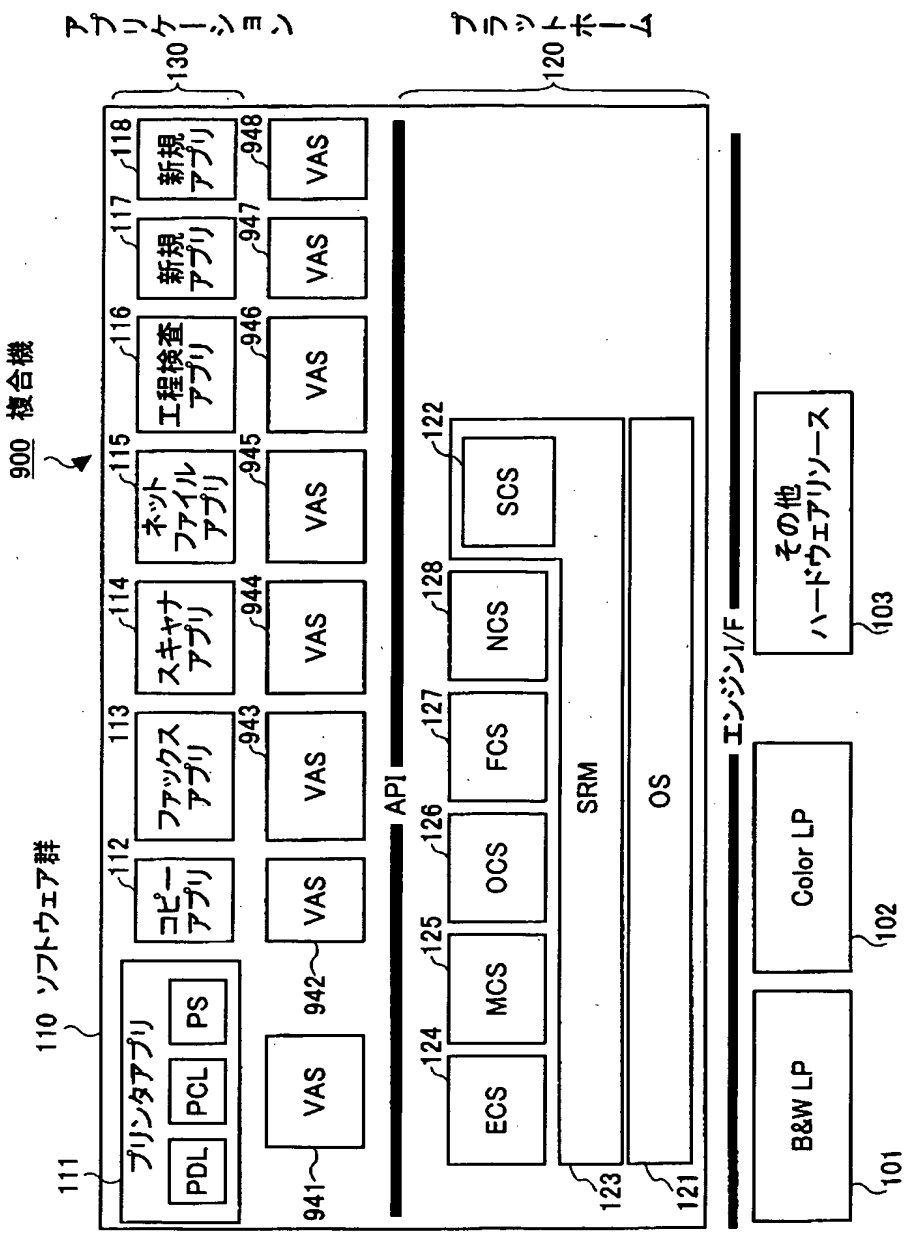
【図１１】

本発明の実施の形態2におけるバージョンチェック処理の手順を示すフローチャート



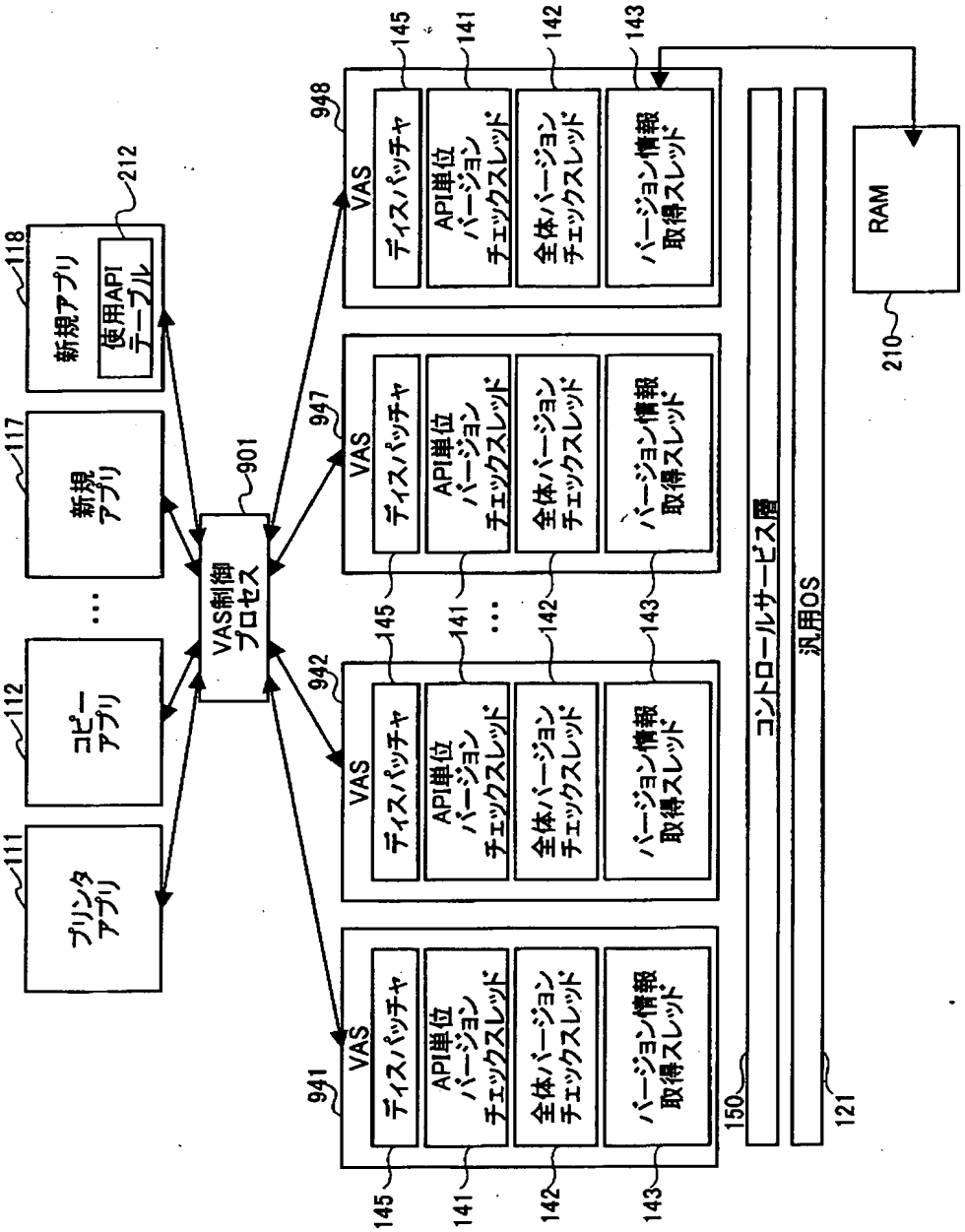
【図１２】

本発明の実施の形態3における複合機のソフトウェア構成を示すブロック図



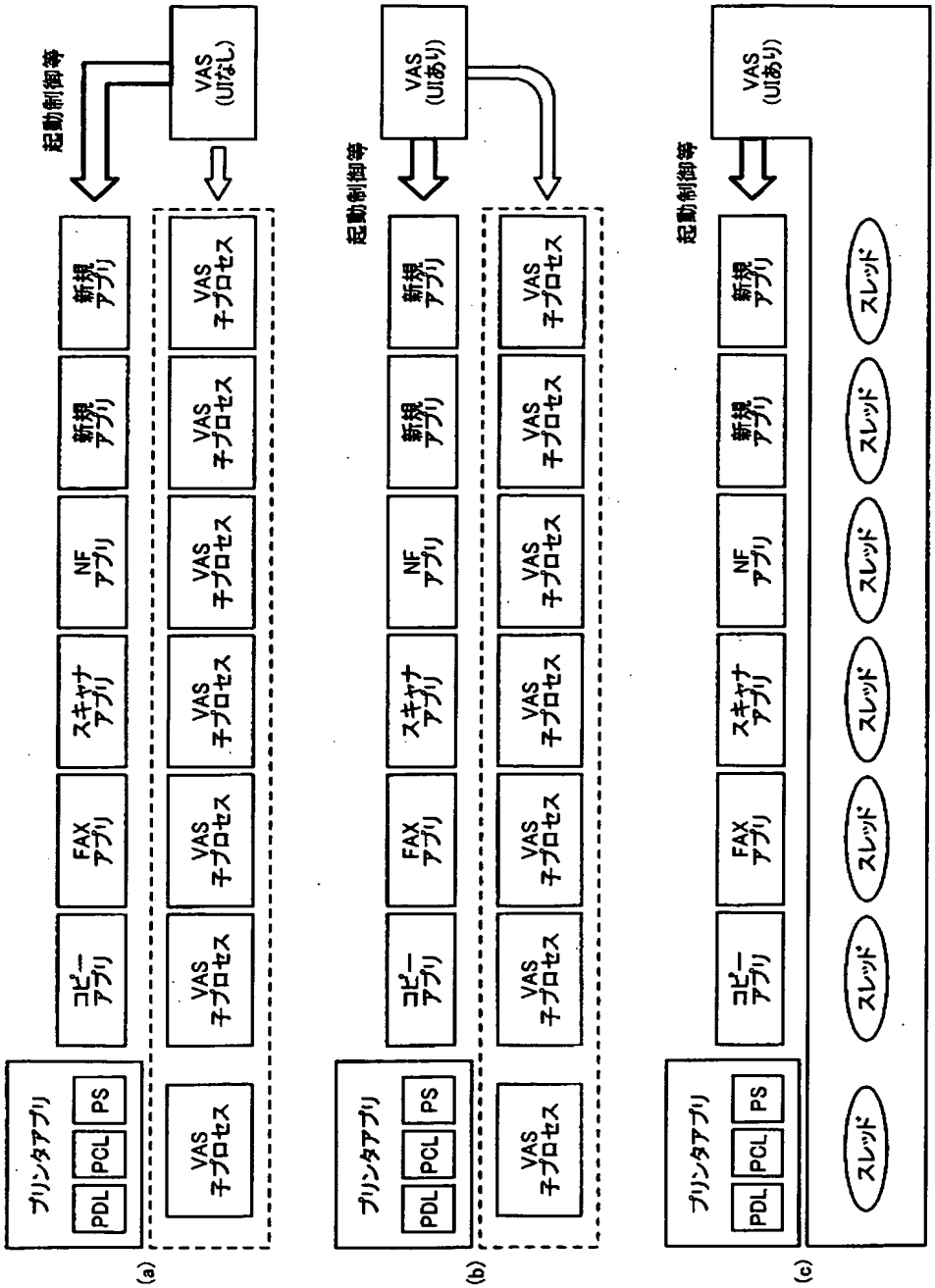
【図１３】

本発明の実施の形態3における複合機のVASの構成と、
VASと各アプリ、コントロールサービス層および
汎用OSとの関係を示すブロック図



【図１４】

本発明の実施の形態におけるVASの構成例を示す図



【書類名】　　要約書

【要約】

【課題】　コントロールサービスなどのシステム側のサービスにおけるＡＰＩのバージョンと、アプリケーションが使用するＡＰＩのバージョンの違いをチェックする。

【解決手段】　アプリケーションと、前記アプリケーションからのＡＰＩを用いた要求に基づきシステム側の処理を行うシステムサービスとを備えた情報処理装置に、前記アプリケーションがシステムサービスに対して使用するＡＰＩのバージョン情報と、当該システムサービスが有するＡＰＩのバージョン情報を取得する取得手段と、前記アプリケーションが前記システムサービスに対して使用するＡＰＩのバージョンを、ＡＰＩ単位に、前記システムサービスが有するＡＰＩのバージョンと比較する比較手段とを備える。

【選択図】　図４

特願２００３－１９５１９４

出　願　人　履　歴　情　報

識別番号　　［０００００６７４７］

１．変更年月日　　１９９０年　８月２４日
［変更理由］　新規登録
住　所　東京都大田区中馬込１丁目３番６号
氏　名　株式会社リコー

２．変更年月日　　２００２年　５月１７日
［変更理由］　住所変更
住　所　東京都大田区中馬込１丁目３番６号
氏　名　株式会社リコー